On calculating guaranteed message response times on the SAE J1939 bus

ROGER JOHANSSON
JAN TORIN

CHALMERS LINDHOLMEN UNIVERSITY COLLEGE
Göteborg, Sweden 2002

Report no.10, Feb. 2002

REPORT No.10

On calculating guaranteed message response times on the SAE J1939 bus

ROGER JOHANSSON[1]
JAN TORIN
Department of Electrical and Computer Engineering
Chalmers Lindholmen University College
roger@chl.chalmers.se

Chalmers Lindholmen University College

Göteborg, Sweden, Feb. 2002

[1] This work has been conducted within the Centre of Excellence CHARMEC (CHAlmers Railway MEChanics) – a VINNOVA Competence Center under the “Programme area 4, SD3. Computer control of braking systems for freight trains”

On calculating guaranteed message response times on the SAE J1939 bus

ROGER JOHANSSON, JAN TORIN

Report – On calculating guaranteed message response times on the SAE J1939 bus

Report no. 10

ISSN 1404-5001

Chalmers Lindholmen University College

P.O. Box 8873

SE-402 72 Göteborg

Sweden

Telephone + 46 31 772 1000

Chalmers Lindholmen University College

Göteborg, Sweden, Feb. 2002

Abstract

Heavy truck manufacturers in automotive applications extensively use the SAE J1939 protocol definition. In this paper we demonstrates two simple methods for analysing worst-case scenarios on a J1939 CAN-bus. The first method can be used for quick estimation of bandwith requirements while the second method illustrates how maximum message delivery time (response time) can be calculated.

1. INTRODUCTION

Since it's introduction in 1995 the SAE J1939 protocol [1] has become a 'de facto' standard for heavy truck manufacturers throughout the world. The recommendation has several benefits; at first, it constitutes a common interface between different control units, supplied by different vendors, on the same bus. Secondly, it provides detailed information about signals (entities) that are communicated on the bus. Thirdly, the recommendations, if accepted and adopted as a *real* standard, may reveal vendors from the difficulties of certification when used in safety critical applications.

The background of this report is part wise the authors experiences gained from coop project with Volvo Truck Corporation in Sweden (VTC) but also a result of joint projects with Saab-Ericsson Space (earlier: Saab Space) and Volvo Car Corporation (VCC). An essential part of chapter 4 is based on earlier published material [4][5].

The major contribution from this report is the derivation of simple, easy to comprehend, equations that describe communication bus utilisation as well as message response latencies in terms of SAE J1939 specification parameters. For all who are working with embedded systems, communicating via a CAN-bus with a J1939 protocol and addressing time critical as well as safety critical application, this report should apply well. The analysis is carried out under the assumptions of fault-free behaviour as well as in the presence of transient faults on the communication bus.

The report is outlined as follows; Chapter 2 gives a short resume of the CAN-protocol. We turn on to a brief description of SAE J1939 protocol concentrating on those fundamental properties of the J1939-protocol that we use in order to derive parameters for our calculations. In chapter 3 we demonstrate some simple derivations of bus utilisation. In chapter 4 we will demonstrate a method of how to compute the worst-case response time of a particular J1939 message. In chapter 5 we summarise our conclusions.

2. THE COMMUNICATION PROTOCOLS

This chapter offers a brief description of the CAN (Controller Area Network) protocol as well as the SAE J1939 vehicle application layer protocol. It is by no mean complete descriptions. The purpose with this chapter is just to provide readers with no previous extensive experiences from these protocols with sufficient knowledge and understanding to comprehend chapters 3 and 4.

Readers with good experiences from CAN and the J1939 protocol might skip this chapter without loss of continuity.

2.1 The CAN protocol

The Controller Area Network (CAN) is a serial communications protocol which supports distributed real-time control applications with dependability requirements. CAN is used in automotive electronics such as engine control modules, transmission control modules, anti brake-lock systems and anti skid systems, just to mention a few, with bit rates up to 1 Mbit/s [1].

The CAN protocol has evolved, from it's initial release in 1986 (also known as *'Standard CAN'*) through the extended protocol definition in 1991 (*'Extended CAN'*, [1]) and is currently under revision for a new specification where, in particularly, time triggered communication issues are addressed (*'Time Triggered CAN'*, TTCAN,[2]).

CAN is a multi-master, broadcast protocol with *collision detect* and a *resolution* mechanism based on message priorities. I.e. each message on the CAN bus has a unique priority and is only transmitted from a single node on the bus.

A CAN message can have one of four different frames:

- A *data* frame carries data from a transmitter to one (or several) receiver(s).
- A node to request a specific data frame transmits a *remote* frame.
- Any receiving node that detects a bus error transmits an *error* frame.
- Any node can transmit an *overload* frame. The purpose with such a frame is to delay the communication.

A data frame is composed of seven different fields (see Table 1 below). The identifier field is used for arbitration in case of contention of the bus. A message with a lower identifier will win any arbitration and lower priority message will have to queue until the bus becomes idle again.

The actual length of any CAN message is always data dependent. The message can contain zero up to eight bytes of data. Furthermore, the number of bits depends on the data itself since the CAN protocol stipulates that a sequence of more than five equal bits has to be broken by insertion of a *stuff* bit to facilitate bit synchronisation by the receiver nodes.

The number of stuff bits may be calculated when we know the message contents; in particular, it is easy to determine the worst case. For example, a CAN Extended data frame has 55 bits of overhead that are subject to bit stuffing. Furthermore, the entire data field is bit stuffed and hence, if we assume a message with *i* data bytes, the maximum number of stuff bits introduced is given by:

$$\left\lfloor \frac{55+8i}{5} \right\rfloor$$

The *End of Frame* and the *Inter Frame Space*, i.e. the bit intervals that must precede a new message on the bus is defined to be total ten bits by the protocol.

	Extended CAN	Subject to bit-stuffing
Start Of Frame	1	X
Arbitration Field	11+2+18+1	X
Control	6	X
Data	64	X
CRC	11	X
CRC delimiter	1	
ACK	2	
EOF/IFS	10	

Table 1 CAN message data frame lengths

The maximum length of an Extended CAN-message (in bits) thus becomes:

$$s = \left\lfloor \frac{55+8i}{5} \right\rfloor + 67 + 8i \qquad \textbf{\textit{Equation 1}}$$

The CAN-protocol error handling mechanism is designed to reach immediate agreement upon disturbed messages. The mechanism works as follow; upon detecting an erroneous CAN-frame, a receiver will destroy the remainder of the message by transmitting an error frame consisting of six dominant bits follow by eight recessive bits. Every node repeats this in the system (see Figure 1 below). The transmitting node, upon detection of the six dominant bits, reinitialises for a retransmission of the frame. CAN-controllers have functions such as error counters designed to achieve fault-tolerance. The subjects of handling errors within controllers are however, far beyond the scope of this report.

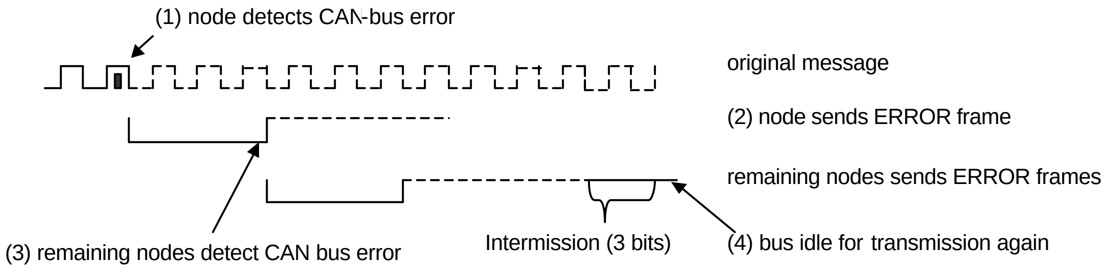
(1) node detects CAN-bus error
original message
(2) node sends ERROR frame
remaining nodes sends ERROR frames
(3) remaining nodes detect CAN bus error
Intermission (3 bits)
(4) bus idle for transmission again

Figure 1 Error handling on the CAN-bus

Consider Figure 1, at (1) any node exhibit an error. As a worst case, let us suppose that no other node detects this error. The node immediately starts transmitting an error frame (2). The latest point where an error is recognised throughout the network is at (3) where the remaining nodes starts sending error frames. An "Intermission" field follows the latest error frame and at (4) latest, error handling are done and the bus is idle for transmission again. Note that in the case where the error was exhibited by all nodes in the system, the scenario is all the same except for that (3) will appear earlier.

We conclude that error handling on the CAN-bus has a bounded time duration determined by the protocol. In the worst case with duration 23 bits (6+6+8+3). The CAN-standard however, stipulates 29 bits of error handling as a worst case when a massive disturbance occurs at the first error flag. We will use this result in chapters 3 and 4.

2.2 SAE J1939 protocol

The J1939 protocol is a vehicle application layer built on top of the CAN-protocol. The central entity is the *Protocol Data Unit* (PDU), which carries all the important information needed for determination of a message's priority and size. The protocol defines all but 8 bits in the CAN identifier, namely the source address field. In fact those eight bits are reserved to uniquely identify a control unit on the bus. Thus there are maximum 256 units allowed on a single J1939-bus. Since no duplicate source id's are allowed on the same bus, this insures that all CAN messages have unique identifiers. It further allows a specific J1939 message to be transmitted from different nodes since they will end up with different CAN identifiers anyway (see Figure 2).

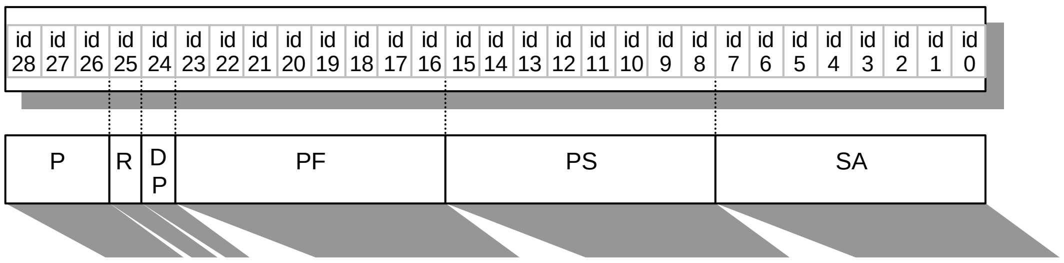

Figure 2 CAN Identifier and the SAE J1939 Protocol Data Unit, bits with identical interpretation are omitted

As shown, different bit fields of the CAN identifier is used for a distinct purpose in a J1939 message. The J1939 interpretation of a CAN identifier is as follows:

- **P**, *priority field*, a 3-bit field allowing priorities 1-8 for a message.
- **R**, *reserved* bit, set to zero by application.
- **DP**, *data page*, address extension (page 1) for *pdu format* (se below).
- **PF**, *pdu format*, specifies a group of messages.
- **PS**, *pdu specific*, used to further determine a specific message, depends on PF.
- **SA**, *source address field*, is the unique signature (0-255) for the electronic control unit that sends the message.

For a specific J1939 message, all fields except the source address field are defined. Thus we conclude that we may establish the necessary database of messages and their properties given that we know all J1939 messages as well as their origins (source identifiers) on a J1939 bus.

We now establish requirements and assumptions that are needed for the development of our analysis.

For the J1939 bus under observation, we require that:
- Bus arbitration, acknowledge and error handling mechanisms obeys the CAN protocol.
- We have established a message database, including all possible J1939 messages, and all control units on the bus.

3. ESTIMATED BUS-UTILISATION

Consider a CAN bus with an arbitrary set of CAN-controllers connected. Assign a unique integer, m, (m = 1,2,3...N) where N is the maximum number of unique messages, to each message transmitted from any of the controllers.

For a message m, with period p_m consisting of s_m bits each of length t, we write the *message bus utilisation* U_m:

$$U_m = \frac{s_m t}{p_m} \qquad \textbf{\textit{Equation 2}}$$

Summarizing all N messages yields the total bus utilisation U ($U \geq 0$):

$$U = t\left(\sum_{m=1}^{N} \frac{s_m}{p_m}\right) \qquad \textbf{\textit{Equation 3}}$$

In the equation we have assumed a *strict* periodic behaviour, which is somewhat unrealistic, therefore let e $(0 \leq e \leq 1)$ denote the fraction of uncertainty in the period time. The actual period of message m now becomes $p_m \pm e_m p_m$. For a worst-case analysis we should throw the '+'-sign since it's not contributes to the worst case. The worst case is obtained if we underestimate the period, that is:

$$p_m (1 - e_m)$$

Which, combined with *Equation 3* yields:

$$U = t\left(\sum_{m=1}^{N} \frac{s_m}{p_m(1-e_m)}\right) \qquad \textbf{\textit{Equation 4}}$$

In the case of disturbances on the bus we might wish to compensate for extra bus utilisation required for error handling in case of disturbed messages. This calculation involves two steps; first we must determine the maximum frequency of disturbed messages, secondly we must calculate the worst-case penalty for the disturbed message. While the first step must be an estimation, the CAN protocol allows us to do a thorough analysis of the second step. Considering activities on the bus we conclude that there are two phases: one, the CAN bus error protocol, i.e. each node sends an error frame upon detecting an invalid message, and two, a retransmission of the message.

The total time for a disturbance T_D , is a sum of the terms:

- T_E, time to do error handling on the CAN-bus
- $T_M = m_{disturbed}\ \{s_m\}\ t$, message transmission time lost due to disturbance

In this approach we have already accounted for retransmission of the message in *Equation 3*.

T_M may be obtained from at least two different approaches; One, we might anticipate the very worst case. i.e. the message which contributes most to U is disturbed. If so we choose the message which yields **max** { U_m } from our set of messages, or secondly, we might anticipate a more stochastic behaviour and choose the message length of **max** {s_m} (based on the assumption that a long message is more likely to be disturbed than a short message. Here, we denote our disturbed message of choice, s_D.

Clearly, the things that determines T_E is *the time when* the message is disturbed as well as the negotiations and handling that occurs upon an erroneous message detected on the bus. The worst case is obtained if is the disturbed message is corrupted in the last bits, as a worst-case assumption we therefore choose s_D here, we then add time for error handling (error frames) and arrive at the results:

$T_E = t\ (s_D + 20)$

$T_M = t\ s_D$

where s_D = [s_m of **max** { U_m } *if worst-case assumed*, **max** { s_m } *otherwise*] (***Definition 1***)

Now assuming our estimated frequency of disturbances is x during a period P_D we write the bus utilisation for disturbances:

$$U_D = x\frac{T_E + T_M}{P_D} \qquad \textbf{\textit{Equation 5}}$$

We now arrive (combining *Equation 4* and *Equation 5*) at our final expression for the bus utilisation:

$$U = t\left[\left(\sum_{m=1}^{N} \frac{s_m}{p_m(1-e_m)}\right) + x\,\frac{2s_D + 20}{P_D}\right] \qquad \textbf{\textit{Equation 6}}$$

where:

U is the maximum bus utilisation for N messages
t is the bit-length time
$\boldsymbol{s_m}$ is the number of bits in message m
$\boldsymbol{p_m}$ is the period of message m
e_m is the maximum fraction of deviation from the stipulated period time of message m
$\boldsymbol{s_D}$ is determined from **Definition 1**, above
$\boldsymbol{P_D}$ is the period time where at most $\boldsymbol{x}$ disturbances are expected.

For ease of comprehension we finally consider a case where all messages are equal in length (s), there is no deviation from message period times and we have a fault free bus. *Equation 6* then simplifies to:

$$U = st\left(\sum_{m=1}^{N}\frac{1}{p_m}\right)$$

I.e. this is a formula, which might be applied upon a realistic set of CAN-bus messages still using a simple hand-held calculator.

4. Response time analysis

The ”response time” of a CAN-message is considered to be the time interval from that the message was eligible for transmission until the time it was acknowledged and thus successfully received by any (i.e. all) other node (nodes) on the CAN-bus. The response time thus constitutes the total occupation of the bus for a successful transmission. The message does not have to be repeated in any sense and will thus not demand any further bus resource. Each successful transmission is also considered as a successful atomic broadcast since the CAN-protocol insures that a single acknowledge guarantees a through out correct reception of the message (otherwise an error frame would have disturbed the message). The “delivery time” of a CAN message is considered to be the time interval from that the message is delivered by an application in a node until it becomes available for other application in other nodes.

For each message we (a priori) must have established:

- A unique priority, i, i.e. the CAN identifier or the J1939 identifier
- Length of the message i (as defined by J1939)
- The i message period time (also defined by J1939)

We consider the CAN-bus to be an *indivisible* resource i.e. once allocated it cannot be shared. (This is most appropriate applied to the CAN-protocol due to its bus arbitration)

We furthermore assume a *constant transmission time*, this is also straightforward since we know each message length s_m as well as the CAN bus baud rate (bits/second).

A message transmission is “non-pre-emptive” i.e. once its started it’s guaranteed to complete (also follows from the CAN-protocol). The schedule is determined by CAN message identifiers a priori and we can use FPS (*Fixed Priority Scheduling*) analysis to compute response times for every CAN message (see [4]).

We now define the total message delivery time D_m for a message m as the time from which it has been delivered by an application to a CAN controller in the sending node, until the message is available for another application in the receiving node. The message delivery time is then contributed to by:

- Time for preparing (formatting) the message for transmission on the CAN-bus.
- Queuing, i.e. waiting time due to lost bus arbitration.
- Transmission time depends on the message length and the bit rate.
- Time for unpacking (de-format) the message and notify the application in the receiver node.

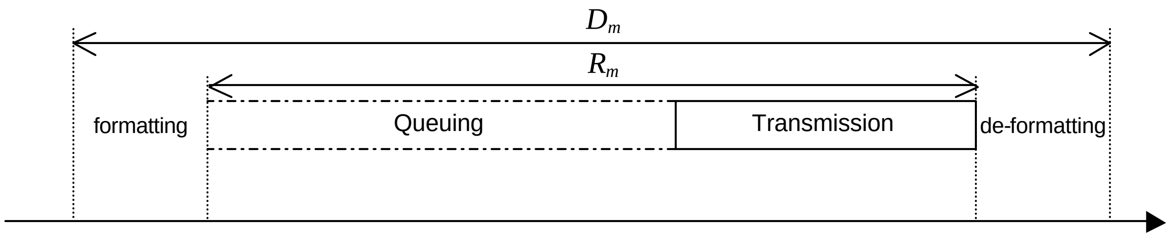

Figure 3 Message delivery time

While the time for preparing and unpacking may be considered constant (depending on the actual CAN controller and operating system) and the transmission time may be calculated for each message, the queuing time depends on the actual schedule. Then, for a message m, in a set (schedule) of N periodic messages ($m = 1..N$) with period p_m consisting of s_m bits each (as determined by *Equation 1*) of length t, the message response time R_m is bounded by (see also [4][5]):

$$R_m = Q_m + T_m$$

where:

- Q_m is the queuing time for message m as a result of higher priority messages transmitted and thus delayed the message m.
- T_m is the transmission time for message m.

Note that messages with $R_m > p_m$ are not guaranteed to be transmitted.

We now turn to the queuing term considering the following example; assume a time interval t. where three messages, m =1,2 and 3, are to be transmitted from different nodes. It can be shown that the worst-case queuing time occurs if all three messages arrive (become eligible for transmission) at the same time (for a proof see [6], Theorem 2). Further, assume that message 1 has the highest priority and message 3 has the lowest priority. Message 1 will not exhibit any delay from queuing since the CAN-bus arbitration mechanism will resolve the bus conflict in favour to message 1.

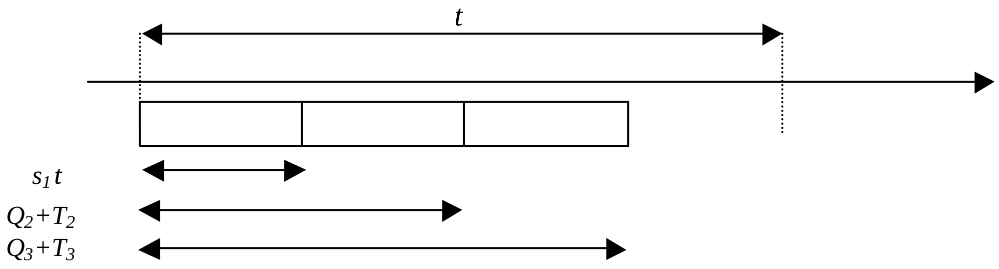

Figure 4 Queuing, awaiting transmission on the CAN bus

In general, the queuing for message *m* is determined by:

$$Q_m = \sum_{\forall j:hp(m)}^{1} \left\lceil \frac{Q_m}{p_j} \right\rceil T_j \qquad \textit{Equation 7}$$

- T_j is the transmission time for a higher priority message *j* and p_j is the period time of a higher priority message *j*.

I.e. during examining of all higher priority messages ($j \subset hp(m)$), where: $Q_1 = 0$, i.e. highest priority message can not be delayed by queuing, *Equation 7* will converge to the solution giving a maximum queuing time.

Due to the non pre-emptive property of a CAN message transmission, **Error! Reference source not found.** does not capture the worst case. We might anticipate a phenomenon called *blocking* [4], which results in an extended queuing delay imposed from a message with lower priority. Consider the following situation where a high priority message is delayed due to such blocking.

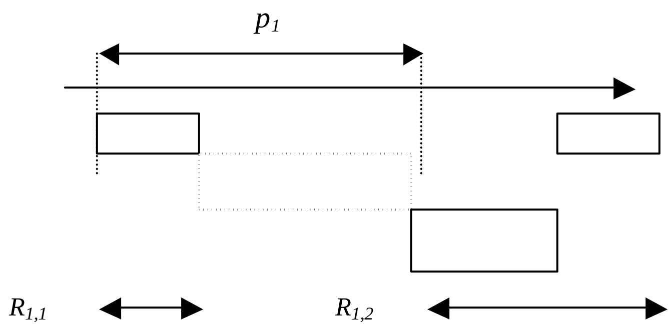

Figure 5 Queuing delay due to blocking

Figure 5 illustrates message blocking. The situation arises because of a low priority message become eligible for transmission just before release of a high priority message. Since the transmission is non pre-emptive, the entire message is transmitted and delays the message delivery time $R_{1,2}$ (second invocation of message 1 transmission). We do however observe that the high priority message can only be blocked by at most one low priority message since during the next bus arbitration, the bus will be allocated to the high priority message, which will be immediately transmitted. Hence, we extend the queuing term with a *blocking term* defined as:

$$B_m = \max\ (T_k) \quad \{\forall k\ lp(m)\ \} \qquad \textit{Equation 8}$$

I.e. The blocking term B_m for message *m* is obtained by examining all lower priority messages and selects the one with greatest transmission time *T*. We now arrive at our final expression for message transmission time:

$$R_m = B_m + T_m + Q_m \qquad \textit{Equation 9}$$

where:

- B_m is the blocking time for message *m* as a result of interference from lower priority messages and is defined by *Equation 8*
- Q_m is the queuing time for message *m* as a result of higher priority messages transmitted and thus delayed the message *m* and defined by *Equation 7*
- T_m is the transmission time for message *m.*

4.1 Robustness

In the analysis we have not accounted for any kind of transmission errors thus far. As a general fault model we might simply assume a disturbed CAN-message that causes other nodes to transmit an error frame. We then further assume that the CAN protocol error handling is obeyed, i.e. the disturbed frame is (eventually) re-transmitted. To establish a satisfactory analysis we must also establish an appropriate fault model. For most cases, it would be naive to expect just a single disturbance. As an example, experiences from measuring error rates on CAN buses in Volvo S80, S60 car models has shown that transmission errors are rare under normal conditions. However, as they occur, one might expect a burst behavior that lasts for a few milliseconds [6]. Consequently, in our response time analysis, we should be able to account for single errors as well as such bursts. Note that we do not address the issues of controller faults but only transmission errors.

4.1.1 Model

- Consider at set of messages, $k, k+1, \ldots N\text{-}1, N$; where k is the highest priority message.
- Construct a time interval T_x where all these messages become eligible for transmission at least once.
- Assume a fault frequency x during the interval T_x.
- As a worst case, assume that the total penalty for a single disturbance equals T_E+T_M as defined in chapter 3.
- Now the term:

$$x\left(\left\lceil \frac{k}{l} \right\rceil - \left\lfloor \frac{k}{l} \right\rfloor\right)(T_E + T_M)$$

Where $k \le l \le N$, and l denotes the disturbed message, accounts for a disturbed message transmission as well as the associated CAN-bus error handling. I.e. we expect no contribution as long as the disturbed message has a lower priority than our message under observation.

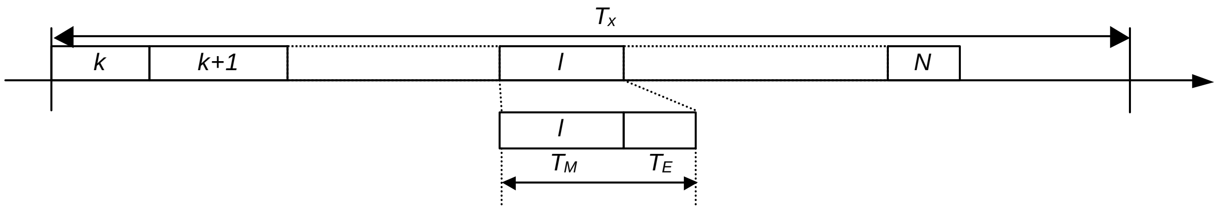

Figure 6 Fault model, delay as result of disturbed message *i*

From the fault model we observe that we has to modify our queuing term for all messages with lower priority than the disturbed message.

$$Q_m = \sum_{\forall j:hp(m)} \left(\left\lceil \frac{Q_m}{p_j} \right\rceil T_j + \sum_{l}^{N} x\left(\left\lceil \frac{j}{l} \right\rceil - \left\lfloor \frac{j}{l} \right\rfloor\right)(T_E + T_M) \right)$$

Equation 10

Figure 6 illustrates the situation. An arbitrary chosen message l scheduled during T_x is disturbed once. We expect penalty in form of lost transmission time (T_M) as well as CAN bus error handling (T_E). These penalties however do not affect messages with higher priority than the disturbed message.

5. CONCLUSIONS

In this paper we have presented ideas for simple but still efficient tools for the analysis of activ ities on a CAN-bus. In particular we concentrated on the SAE J1939 vehicle application layer.

In chapter 3 we presented, easy to use, formulas that might provide a range of useful information. For example, by monitoring (or analysing) the messages that are transmitted from a particular control unit (CAN-node) one might get an opinion about the total bus bandwith required by that unit. Furthermore, by examining worst-case scenarios one might predict troublesome situations in a system. For example by extensively studying situations where the bus utilisation is above 1, i.e. where messages definitively is lost. In chapter 4 we give a receipt for those situations. By applying the methods presented in chapter 4 it is possible to find the messages that will suffer from an overloaded bus, i.e. the messages that will exhibit (perhaps substantially) jitter in their period.

While the method presented in chapter 3 is easy to carry out, even on a pocket calculator, the method that is presented in chapter 4 might be more cumbersome to implement. A rudimentary implementation has been done within this work. A more serious approach probably requires more efforts, such as systemising the J1939 messages in a database and developing tools that extracts and uses information from this database.

ACKNOWLEDGES

The authors wish to thank Lars Berno Fredriksson at KVASER for his comments and suggestions.

REFERENCES

[1] Society of Automotive Engineers, SAE, *Surface Vehicle Recommended Practice J1939-7x*.

[2] Robert Bosch GmbH, *CAN Specification Version 2.0*, 1991.

[3] ISO/WD 11898-4, *Road vehicles – Controller area network (CAN) – Part 4: Time triggered communication*, December 2000.

[4] Tindell, K., Burns, A., Wellings A.J., *Calculating Controller Area Network (CAN) Message Response Times, Software Engineering Journal*, 1995.

[5] Sha, L.,Rajkumar, R.,Sathaye, S., *Generalized Rate-Monotonic Scheduling Theory: A Framework for Developing Real-Time Systems*. Proceedings of the IEEE, Vol 82, No 1, January 1994, pp68-82.

[6] Ericson, R., Hagardzon, H., *CAN-bus error counter (CANEC)*, Bach.Thesis, Chalmers Lindholmen University College, May 2001.